

WHAT IS CLAIMED IS:

- 1 1. A computer-implemented method of identifying
2 compatible software threads to execute on an SMT
3 processor, said method comprising:
4 identifying a time interval during which both a first
5 thread and a second thread are executing on the SMT
6 processor;
7 retrieving a performance value that occurred during
8 the identified time interval;
9 determining, based upon the retrieved performance
10 value, whether the first thread is compatible with the
11 second thread; and
12 recording the compatibility of the first thread with
13 the second thread in response to the determination.
- 1 2. The method as described in claim 1 wherein the
2 performance value is a cycles per instruction (CPI)
3 value.
- 1 3. The method as described in claim 2 wherein the CPI
2 value further comprises:
3 retrieving a number of cycles value indicating the
4 number of processing cycles that occurred during the
5 time interval;
6 retrieving a number of instructions value indicating
7 the number of instructions that were executed by the
8 SMT processor during the time interval; and

9 dividing the number of cycles value by the number of
10 instructions value, the dividing resulting in the CPI
11 value.

1 4. The method as described in claim 2 further comprising:
2 comparing the CPI value to a threshold value, wherein
3 the first thread and second thread are determined to
4 be compatible if the CPI value is better than the
5 threshold value.

1 5. The method as described in claim 2 further comprising:
2 writing a first identifier corresponding to the first
3 thread and the CPI value to a compatibility list that
4 corresponds to the second thread.

1 6. The method as described in claim 5 wherein the writing
2 is performed in response to identifying an empty field
3 in the second thread's compatibility list.

1 7. The method as described in claim 5 wherein the writing
2 is performed in response to:
3 comparing the CPI value to one or more previously
4 recorded CPI values that correspond to one or more
5 previously identified compatible threads; and
6 determining that the CPI value is better than at least
7 one of the previously recorded CPI values.

1 8. The method as described in claim 7 further comprising:

2 removing one of the previously recorded CPI values and
3 data corresponding to one of the previously identified
4 compatible threads prior to the writing.

1 9. The method as described in claim 1 further comprising:
2 writing a first identifier corresponding to the first
3 thread to a compatibility list corresponding to the
4 second thread, wherein the compatibility list stores a
5 plurality of thread identifiers compatible with the
6 second thread.

1 10. The method as described in claim 9 further comprising:
2 writing a timestamp corresponding to the first
3 identifier, the timestamp indicating a time at which
4 the time interval occurred, wherein each of the
5 plurality of thread identifiers also include a
6 plurality of timestamps indicating when each of the
7 threads executed with the second thread.

1 11. The method as described in claim 10 further
2 comprising:

3 periodically cleaning a plurality of compatibility
4 lists, including the second thread's compatibility
5 list, the cleaning including:

6 reading the entries corresponding to each of the
7 threads listed in the compatibility lists;

8 comparing the timestamps listed in the
9 compatibility list with a current time;

10 determining, based on the comparison, whether the
11 entry associated with the timestamp is stale; and
12 removing the entry in response to determining
13 that it is a stale entry.

1 12. The method as described in claim 1 further comprising:
2 sensing that either the first thread or the second
3 thread is about to complete;
4 scheduling a new thread to execute, the scheduling
5 comprising:
6 identifying a compatible thread, the compatible
7 thread being compatible to the tread that is not
8 about to complete;
9 determining whether the compatible thread is
10 ready to execute; and
11 dispatching the compatible thread to execute on
12 the SMT processor.

1 13. The method as described in claim 12 wherein the thread
2 that is about to complete and the compatible thread
3 are listed in a first run queue and wherein the thread
4 that is not about to complete is listed in a second
5 run queue.

1 14. A computer-implemented method of dispatching software
2 threads to execute on an SMT processor, said method
3 comprising:
4 sensing that a completing thread that is about to
5 complete execution on the SMT processor;

6 identifying a running thread that is still executing
7 on the SMT processor;

8 checking a list of one or more compatible threads,
9 wherein the compatible threads are compatible with the
10 running thread;

11 determining that one of the compatible threads is
12 ready to execute; and

13 dispatching the determined thread to execute on the
14 SMT processor.

1 15. The method as described in claim 14 wherein the
2 completing thread and the compatible threads are
3 listed in a first run queue and wherein the running
4 thread is listed in a second run queue.

1 16. The method as described in claim 14 wherein the
2 determination that one of the compatible threads is
3 ready to execute further comprises:
4 checking whether the compatible threads are ready to
5 execute in order of a CPI value, wherein the
6 compatible threads are checked in an order determined
7 by a CPI value corresponding to each of the compatible
8 threads, so that compatible threads with better CPI
9 values are checked first.

1 17. An information handling system comprising:
2 one or more SMT processors;
3 a memory accessible by the processors;

4 a compatibility tool for identifying compatible
5 threads to execute on one of the SMT processors, the
6 compatibility tool comprising software code effective
7 to:

8 identify a time interval during which both a
9 first thread and a second thread are executing on
10 the SMT processor;

11 retrieve a performance value that occurred during
12 the identified time interval;

13 determine, based upon the retrieved performance
14 value, whether the first thread is compatible to
15 the second thread; and

16 record the compatibility of the first thread to
17 the second thread in response to the
18 determination.

1 18. The information handling system as described in claim
2 17 wherein the performance value is a cycles per
3 instruction (CPI) value.

1 19. The information handling system as described in claim
2 18, wherein the CPI value is computed using software
3 code effective to:

4 retrieve a number of cycles value indicating the
5 number of cycles that occurred during the time
6 interval;

7 retrieve a number of instructions value indicating the
8 number of instructions that were executed during the
9 time interval; and

10 divide the number of cycles value by the number of
11 instructions value, the dividing resulting in the CPI
12 value.

1 20. The information handling system as described in claim
2 18 further comprising software code effective to:
3 compare the CPI value to a threshold value, wherein
4 the first thread and second thread are determined to
5 be compatible if the CPI value is better than the
6 threshold value.

1 21. The information handling system as described in claim
2 18 further comprising software code effective to:
3 write a first identifier corresponding to the first
4 thread and the CPI value to a compatibility list that
5 corresponds to the second thread.

1 22. The information handling system as described in claim
2 21 wherein the writing is performed in response to
3 software code effective to identify an empty field in
4 the second thread's compatibility list.

1 23. The information handling system as described in claim
2 21 wherein the writing is performed in response to
3 software code effective to:
4 compare the CPI value to one or more previously
5 recorded CPI values that correspond to one or more
6 previously identified compatible threads; and
7 determine that the CPI value is better than at least
8 one of the previously recorded CPI values.

1 24. The information handling system as described in claim
2 23 further comprising software code effective to:
3 remove one of the previously recorded CPI values and
4 data corresponding to one of the previously identified
5 compatible threads prior to the writing.

1 25. The information handling system as described in claim
2 17 further comprising software code effective to:
3 write a first identifier corresponding to the first
4 thread to a compatibility list corresponding to the
5 second thread, wherein the compatibility list stores a
6 plurality of thread identifiers compatible with the
7 second thread.

1 26. The information handling system as described in claim
2 25 further comprising software code effective to:
3 write a timestamp corresponding to the first
4 identifier, the timestamp indicating a time at which
5 the time interval occurred, wherein each of the
6 plurality of thread identifiers also include a
7 plurality of timestamps indicating when each of the
8 threads executed with the second thread.

1 27. The information handling system as described in claim
2 26 further comprising software code effective to:
3 periodically clean a plurality of compatibility lists,
4 including the second thread's compatibility list, the
5 cleaning comprising software code effective to:

6 read the entries corresponding to each of the
7 threads listed in the compatibility lists;
8 compare the timestamps listed in the
9 compatibility list with a current time;
10 determine, based on the comparison, whether the
11 entry associated with the timestamp is stale; and
12 remove the entry in response to determining that
13 it is a stale entry.

1 28. The information handling system as described in claim
2 17 further comprising software code effective to:
3 sense that either the first thread or the second
4 thread is about to complete;
5 schedule a new thread to execute, the scheduling
6 comprising software code effective to:
7 identify a compatible thread, the compatible
8 thread being compatible to the tread that is not
9 about to complete;
10 determine whether the compatible thread is ready
11 to execute; and
12 dispatch the compatible thread to execute on the
13 SMT processor.

1 29. The information handling system as described in claim
2 28 wherein the thread that is about to complete and
3 the compatible thread are listed in a first run queue
4 and wherein the thread that is not about to complete
5 is listed in a second run queue.

1 30. An information handling system comprising:
2 one or more SMT processors;
3 a memory accessible by the processors;
4 a dispatching tool for dispatching compatible threads
5 to execute simultaneously on one of the SMT
6 processors, the dispatching tool comprising software
7 code effective to:
8 sense that a completing thread that is about to
9 complete execution on the SMT processor;
10 identify a running thread that is still executing
11 on the SMT processor;
12 check a list of one or more compatible threads,
13 wherein the compatible threads are compatible
14 with the running thread;
15 determine that one of the compatible threads is
16 ready to execute; and
17 dispatch the determined thread to execute on the
18 SMT processor.

1 31. The information handling system as described in claim
2 30 wherein the completing thread and the compatible
3 threads are listed in a first run queue and wherein
4 the running thread is listed in a second run queue.

1 32. The information handling system as described in claim
2 30 wherein the determination that one of the
3 compatible threads is ready to execute further
4 comprises software code effective to:

5 check whether the compatible threads are ready to
6 execute in order of a CPI value, wherein the
7 compatible threads are checked in an order determined
8 by a CPI value corresponding to each of the compatible
9 threads, so that compatible threads with better CPI
10 values are checked first.

1 33. A computer program product stored on a computer
2 operable media for identifying compatible software
3 threads to execute on an SMT processor, said computer
4 program product comprising:

5 means for identifying a time interval during which
6 both a first thread and a second thread are executing
7 on the SMT processor;

8 means for retrieving a performance value that occurred
9 during the identified time interval;

10 means for determining, based upon the retrieved
11 performance value, whether the first thread is
12 compatible with the second thread; and

13 means for recording the compatibility of the first
14 thread with the second thread in response to the
15 determination.

1 34. The computer program product as described in claim 33
2 wherein the performance value is a cycles per
3 instruction (CPI) value.

1 35. The computer program product as described in claim 34
2 wherein computing the CPI value comprises:

means for retrieving a number of cycles value indicating the number of processing cycles that occurred during the time interval;

means for retrieving a number of instructions value indicating the number of instructions that were executed by the SMT processor during the time interval; and

means for dividing the number of cycles value by the number of instructions value, the dividing resulting in the CPI value.

36. The computer program product as described in claim 34 further comprising:

means for comparing the CPI value to a threshold value, wherein the first thread and second thread are determined to be compatible if the CPI value is better than the threshold value.

37. The computer program product as described in claim 34 further comprising:

means for writing a first identifier corresponding to the first thread and the CPI value to a compatibility list that corresponds to the second thread.

38. The computer program product as described in claim 37 wherein the writing is performed in response to identifying an empty field in the second thread's compatibility list.

1 39. The computer program product as described in claim 37
2 wherein the means for writing is performed in response
3 to:

4 means for comparing the CPI value to one or more
5 previously recorded CPI values that correspond to one
6 or more previously identified compatible threads; and

7 means for determining that the CPI value is better
8 than at least one of the previously recorded CPI
9 values.

1 40. The computer program product as described in claim 39
2 further comprising:

3 means for removing one of the previously recorded CPI
4 values and data corresponding to one of the previously
5 identified compatible threads prior to the writing.

1 41. The computer program product as described in claim 33
2 further comprising:

3 means for writing a first identifier corresponding to
4 the first thread to a compatibility list corresponding
5 to the second thread, wherein the compatibility list
6 stores a plurality of thread identifiers compatible
7 with the second thread.

1 42. The computer program product as described in claim 41
2 further comprising:

3 means for writing a timestamp corresponding to the
4 first identifier, the timestamp indicating a time at
5 which the time interval occurred, wherein each of the

6 plurality of thread identifiers also include a
7 plurality of timestamps indicating when each of the
8 threads executed with the second thread.

1 43. The computer program product as described in claim 42
2 further comprising:

3 means for periodically cleaning a plurality of
4 compatibility lists, including the second thread's
5 compatibility list, the cleaning including:

6 means for reading the entries corresponding to
7 each of the threads listed in the compatibility
8 lists;

9 means for comparing the timestamps listed in the
10 compatibility list with a current time;

11 means for determining, based on the comparison,
12 whether the entry associated with the timestamp
13 is stale; and

14 means for removing the entry in response to
15 determining that it is a stale entry.

1 44. The computer program product as described in claim 33
2 further comprising:

3 means for sensing that either the first thread or the
4 second thread is about to complete;

5 means for scheduling a new thread to execute, the
6 scheduling comprising:

7 means for identifying a compatible thread, the
8 compatible thread being compatible to the tread
9 that is not about to complete;

10 means for determining whether the compatible
11 thread is ready to execute; and
12 means for dispatching the compatible thread to
13 execute on the SMT processor.

1 45. The computer program product as described in claim 44
2 wherein the thread that is about to complete and the
3 compatible thread are listed in a first run queue and
4 wherein the thread that is not about to complete is
5 listed in a second run queue.

1 46. A computer program product stored on a computer
2 operable media for dispatching software threads to
3 execute on an SMT processor, said computer program
4 product comprising:
5 means for sensing that a completing thread that is
6 about to complete execution on the SMT processor;
7 means for identifying a running thread that is still
8 executing on the SMT processor;
9 means for checking a list of one or more compatible
10 threads, wherein the compatible threads are compatible
11 with the running thread;
12 means for determining that one of the compatible
13 threads is ready to execute; and
14 means for dispatching the determined thread to execute
15 on the SMT processor.

1 47. The computer program product as described in claim 46
2 wherein the completing thread and the compatible

3 threads are listed in a first run queue and wherein
4 the running thread is listed in a second run queue.

1 48. The computer program product as described in claim 46
2 wherein the determination that one of the compatible
3 threads is ready to execute further comprises:

4 means for checking whether the compatible threads are
5 ready to execute in order of a CPI value, wherein the
6 compatible threads are checked in an order determined
7 by a CPI value corresponding to each of the compatible
8 threads, so that compatible threads with better CPI
9 values are checked first.